UltraScale Bitstreams Lab

Introduction

This lab demonstrates:

1. Implementing a simple design using Vivado project mode
2. Writing bitstreams using Tcl commands and options not supported by project mode
3. Writing a bitstream that uses an obfuscated AES-256 key
4. Writing a bitstream that uses a finite key life (i.e. key rolling)
5. Query bitstream properties

This lab uses the following steps:

1. Create a Vivado RTL project
2. Implement the RTL project
3. Create a bitstream that uses an obfuscated AES-256 key
4. Create a bitstream that uses key rolling.
5. Query bitstream properties

Objectives

After this lab you will be able to do the following

- Create a Vivado RTL project
- Implement the Vivado RTL project
- Write bitstreams using various options
- Query bitstream properties

Prerequisites

This lab is separated into 5 steps that consist of general overview statements that provide information on the detailed instructions that follow.

The lab expects the following prerequisites:

I. Vivado/SDK 2014.3 installed on your host machine.
The user should have valid license for the tools to run the lab.

2 The HxD hex editor (http://mh-nexus.de/en/hxd/) is used to examine bit files. Another hex editor would also be acceptable.

Creating the Vivado RTL Project Step 1

This step walks through creating the Vivado RTL Project. Source code required for the project is already installed at C:/Projects/2014_XSWG_Bitstream_Lab.

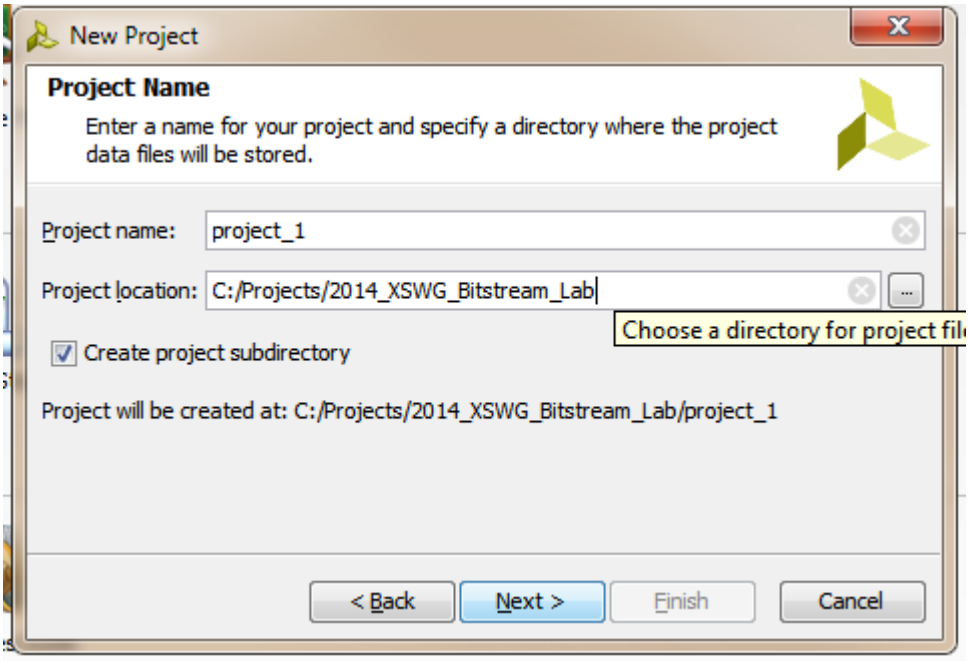
1-1. The required support files for this lab are present in the 2104_XSWG_Bistream_Lab.zip file. Unzip the contents of the zip file into your C:\Projects.

1-2. Launch Vivado 2014.3.

1-3. Use the “Create New Project” Quick Start button to launch the project wizard.

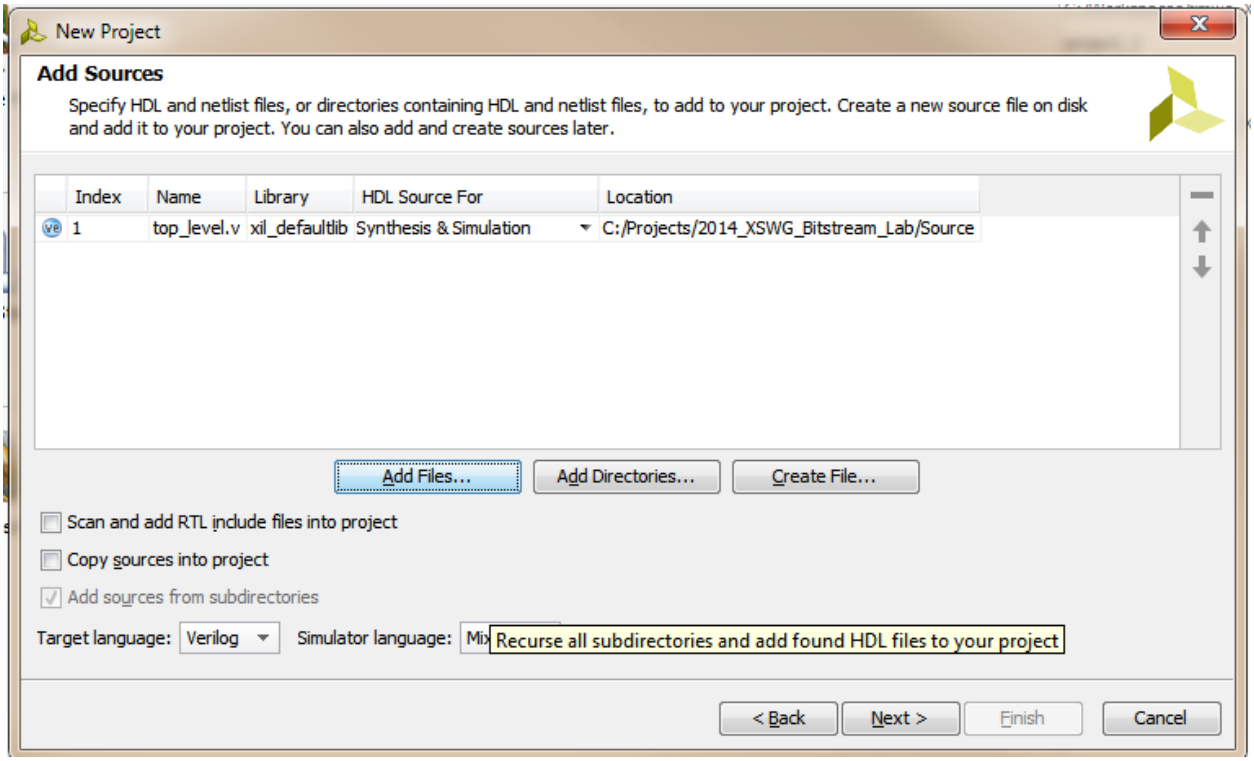
1-3-1. In the Create a New Vivado Project Window, click **Next**.

1-3-2. In the Project Name window, select a Project Name and Project location. This example will use project_1 and C:\Projects\2014_XSWG_Bitstream_Lab. Click **next**.



1-3-3. In the Project Type window, leave the RTL Project radio button selected and click **next**.

1-3-4. In the Add Sources window, select **Add Files**. Browse to the Source directory and select top_level.v. Click **OK**. When the Add Sources window returns, click **next**.



1-3-5. In the Add Existing IP (optional) window, click **next.**

1-3-6. In the Add Constraints (optional) window, click **Add files**. Select the top_level.xdc file and click **OK.** When the Add Constraints (optional) window returns, click **Next.**

1-3-7. In the Default Part window, select the UltraScale part xcku040-ffva1156-1-c part. Click **Next.**

1-3-8. In the New Project Summary window, click **Finish.**

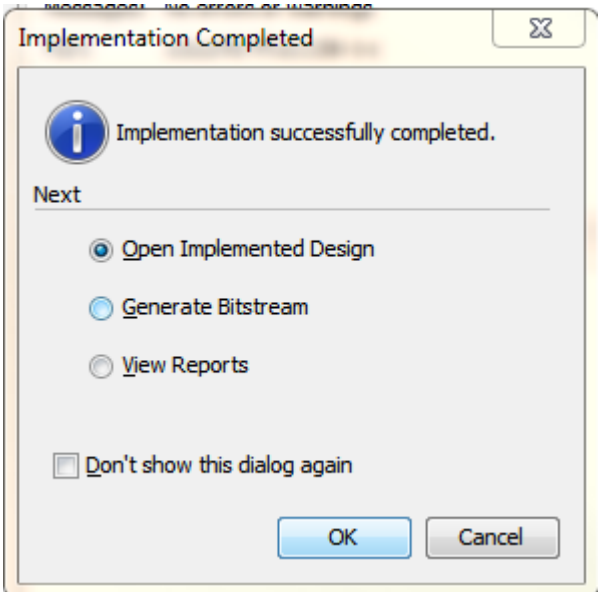
Implementing the Vivado Project — Step 2

This step walks through implementing the Vivado project.

2-1. In the Project Manager window, click Run Implementation.

2-2. When the Missing Synthesis Runs window appears asking if synthesis should be launched, click OK.

2-3. Once the Implementation Completed window appears, leave the Open Implemented Design radio button selected and click OK.



Create the bitstream using an obfuscated key Step 3

The UltraScale devices have the ability to decrypt and obfuscated key using a metal key implemented in the silicon. This metal key can be used to encrypt or obfuscate the user key before so that the true user key doesn't need to be provided to a contract manufacturer. The contract manufacture can program the obfuscated key into the EFUSEs or BBRAM and the device will decrypt it in order to recover the user key.

Now that the project is implemented **and opened**, Tcl commands can be used to write the bitstream with the desired bitstream options. The required bitstream options aren't supported in the GUI so they will be set manually using the **set_property** Tcl command.

3-1. Select the Tcl Console tab.

3-2. In the Tcl Console, type the following commands**.**

3-2-1. To enable encryption

```
set_property bitstream.encryption.encrypt yes [current_design]
```

3-2-2. To select EFUSE key storage (as opposed to battery-backed RAM (BBR) key storage)

```
set_property bitstream.encryption.encryptKeySelect efuse [current_design]
```

3-2-3. To select the key file containing the desired customer AES-256 key

set_property bitstream.encryption.keyFile
c:/Projects/2014_XSWG_Bitstream_Lab/Source/myKey.nky [current_design]

3-2-1. To enable a bitstream that will use an obfuscated EFUSE key

set_property bitstream.encryption.obfuscateKey enable [current_design]

3-2-1. To write the bitstream

write_bitstream –force c:/Projects/2014_XSWG_Bitstream_Lab/obfuscatedKey.bit

3-3. To find the obfuscated key that can be given to a contract manufacturer for programming into the device's EFUSE, open the file c:/Projects/2014_XSWG_Bitstream_Lab/obfuscatedKey.nky**.** This file was created by the write_bitstream command along with the bit file.

```
Device xcku040;
EncryptKeySelect EFUSE;
KeyObfuscate da2ac90eb692e144e192e8bae1f18a60eb754b75e82009fb611c399149434ee7;
Key0 f878b838d8589818e868a828c8488808f070b030d0509010e060a020c0408000;
StartIV0 5c9d95ecbfec8a1f12a8eb312362c596;
```

The KeyObfuscate field contains the obfuscated result of encrypting Key0 with the metal key.

3-4. Examine the bitstream using HxD or another hex editor. The following screen shot was made by deleting all the hex values prior to the start of the 0xFFFF_FFFF preamble and then changing the **View->Byte group size** to 4.

Create the bitstream using key rolling — Step 4

The AES-256 key needs to be protected against DPA attacks. One of the protection mechanisms is to limit the amount of data that is decrypted with any one key. In order to do this, write_bitstream can create a bitstream that changes the key every so many encryption blocks. This limits the amount of data encrypted on any one key. This feature is called key rolling.

With the implemented project still open, Tcl commands can be used to write another bitstream with the desired bitstream options. The required bitstream options aren't

supported in the GUI so they will be set manually using the **set_property** Tcl command. Options from the prior step are still being applied by the tool.

4-1. In the Tcl Console, type the following commands**.**

4-1-1. To disable key obfuscation

set_property bitstream.encryption.obfuscateKey disable [current_design]

4-1-2. To select the key file containing the desired customer AES-256 key. This key file defines the first 100 keys. It could be modified to define all the required keys. The tool will create more keys if needed.

set_property bitstream.encryption.keyFile
c:/Projects/2014_XSWG_Bitstream_Lab/Source/my100Keys.nky [current_design]

4-1-3. To set the keyLife to 200 encryption blocks.

set_property bitstream.encryption.keyLife 200 [current_design]

4-1-4. To write the bitstream

write_bitstream –force c:/Projects/2014_XSWG_Bitstream_Lab/keyRolling.bit

4-2. Open the file c:/Projects/2014_XSWG_Bitstream_Lab/keyRolling.nky**.** This file was created by the write_bitstream command along with the bit file. Notice that it contains over 5000 keys even though the original key file only specified the first 100.

4-3. Examine the bitstream using HxD or another hex editor. Below is a screen capture of a portion of the bistream between the 1st and 2nd encrypted messages. The following screen shot was made by deleting all the hex values prior to the start of the 0xFFFF_FFFF preamble and then changing the **View->Byte group size** to 4.

```
00000D70  3C6F5809 2DBE853C ECD2D410 36BA6B75  <oX.-¾…<ìÒÔ.6°ku
00000D80  8898498B 57CA81FD FE0EB93E 6E354242  ˆ˜I‹WÊ.ýþ.¹>n5BB
00000D90  C3DF2475 A0624441 56C53104 2F7B6BF8  Ãß$u bDAVÅ1./{kø
00000DA0  A8099293 A36F7714 747F3CBF 89B6A1C6  ¨.'"£ow.t.<¿‰¶¡Æ
00000DB0  18805FE6 A88249FE 94187594 2BF63BBD  .€_æ¨,Iþ".u"+ö;½
00000DC0  8C95E910 707B55A5 9B76CB6B 56095525  Œ•é.p{U¥›vËkV.U%
00000DD0  6DC34D94 8E654EE0 EC21329C EC8143F2  mÃM"ŽeNài!2œì.Cò
00000DE0  AE6CCF35 77FEFEFB 6ED68D3A C2ACACDC  ®lÏ5wþþûnÖ.:Â¬¬Ü
00000DF0  65E50258 FF538831 9ABAB947 FA16C4DC  eå.XÿS8ˆ1š°¹Gú.ÄÜ
00000E00  20000000 20000000 20000000 20000000    ... ... ... ...
00000E10  20000000 20000000 20000000 20000000    ... ... ... ...
00000E20  20000000 20000000 20000000 20000000    ... ... ... ...
00000E30  20000000 20000000 20000000 20000000    ... ... ... ...
00000E40  20000000 20000000 20000000 20000000    ... ... ... ...
00000E50  20000000 20000000 20000000 20000000    ... ... ... ...
00000E60  20000000 20000000 20000000 20000000    ... ... ... ...
00000E70  20000000 20000000 20000000 20000000    ... ... ... ...
00000E80  30016004 40C4B26C E6A0935E 0B02853D  0.`.@Ä²læ "^..…=
00000E90  00000324 20000000 20000000 20000000  ...$ ... ... ...
00000EA0  20000000 20000000 20000000 20000000    ... ... ... ...
00000EB0  20000000 20000000 20000000 20000000    ... ... ... ...
00000EC0  20000000 20000000 20000000 20000000    ... ... ... ...
00000ED0  20000000 20000000 20000000 20000000    ... ... ... ...
00000EE0  20000000 20000000 20000000 20000000    ... ... ... ...
00000EF0  20000000 20000000 20000000 20000000    ... ... ... ...
00000F00  20000000 20000000 20000000 20000000    ... ... ... ...
00000F10  20000000 20000000 20000000 20000000    ... ... ... ...
00000F20  20000000 20000000 20000000 20000000    ... ... ... ...
00000F30  20000000 20000000 20000000 20000000    ... ... ... ...
00000F40  20000000 20000000 20000000 20000000    ... ... ... ...
00000F50  20000000 20000000 20000000 20000000    ... ... ... ...
00000F60  20000000 20000000 20000000 20000000    ... ... ... ...
00000F70  20000000 20000000 20000000 20000000    ... ... ... ...
00000F80  20000000 2F94D2C8 1E22EEFD 32B79B28    .../"ÒÈ."îý2·›(
00000F90  8DC24011 7D082237 53F776B9 B4FAF5DB  .Â@.}."7S÷v¹´úõÛ
00000FA0  9FDEC2E5 F2614EEA F302B746 B0E2DFB3  ŸÞÂåòaNêó.·F°âß³
00000FB0  815F2DBE 2F594013 64597F96 5255F30F  ._-¾/Y@.dY.-RUó.
00000FC0  B428EADC 689B9D8D 3E47D48D 9D7AA8A3  ´(êÜh›..>GÔ..z¨£
00000FD0  496AFAAD 2538AD4C 716D9536 B03463AE  Ijú.%8.Lqm•6°4c®
00000FE0  7478101E F44547A7 A62B4BB2 A5192924  tx..ôEG§¦+K²¥.)$
00000FF0  431A3659 382B757C 41DFBD76 37B4C0B1  C.6Y8+u|AßŠv7´À±
00001000  CF3F8464 B7BF64C1 99C3903D F51B4988  Ï?d·¿dÁ™Ã.=õ.I˜
```

End of the 1st encrypted message using the key in EFUSES or BBR

IV of the 2nd message

Start of the 2nd encrypted message using the key that was embedded in the 1st encrypted message

Querying bitstream properties — Step 5

Since not all bitstream properties are supported by the project mode GUI, it is important to be able to query the status of bitstream properties that have been set and to query which bitstream properites are available.

5-1. To report the bitstream properties that are set, type the following Tcl command

report_property [current_design] –regexp BITSTREAM.*

5-2. To report all available bitstream properties, type the following Tcl command

report_property –all [current_design] –regexp BITSTREAM.*

Support for RSA Authenticated bitstreams will be available in 2014.4. Specifying an RSA Authenticated bitstream will use the BITSTREAM.* properties as well.